Specification

Title of the Invention

Program Development Support Apparatus

Background of the Invention

The present invention relates to a program development support apparatus and, more particularly, to a program development support apparatus which supports debug of a computer program.

A program development support apparatus is especially used to develop and debug a program for operating a computer apparatus. Particularly, a program development support apparatus having a trace function is used to store an instruction address and instruction code of a CPU (Central Processing Unit), which generally change momentarily as a program is executed, in a memory called a trace memory and analyze them as a program execution result.

In recent years, programs installed in computer apparatuses are becoming complex and bulky, and a trace memory is required to have a large capacity to store a larger amount of execution result. On the other hand, since the operation speed of a CPU increases, a faster memory is necessary. Generally, a program development support apparatus having a trace function need be operated at the same speed as the operation frequency of the CPU. However, since a large-capacity,

high-speed memory is very expensive, a decrease in amount of trace data to be stored in the trace memory has been required.

A technique that meets this requirement is disclosed in Japanese Patent Laid-Open No. 11-259335 (reference 1), in which the difference value between the immediately preceding program counter value and the current program counter value is recorded on the trace memory, and only for a branch instruction, the program counter value itself is recorded, thereby compressing the trace data. Fig. 12 shows the conventional program development support apparatus disclosed in reference 1.

Referring to Fig. 12, the program development support apparatus has an evaluation chip 301 prepared for program development and a tracer 302 for storing an instruction trace result. Instruction address/instruction code data output from a CPU 303 in the evaluation chip 301 is latched by an instruction address/instruction code latch circuit 321 in synchronism with a clock signal CLK, and a branch instruction determination circuit 323 determines whether the instruction is a branch instruction.

When the instruction is not a branch instruction, an instruction address data compression circuit 322 compresses the instruction address by generating the difference value between the immediately preceding program counter value and the current program

counter value. A trace control circuit 324 combines compressed data of a plurality of instruction addresses into trace data in accordance with the bit width of a trace memory 306. The combined trace data is written in the trace memory 306. If the instruction is a branch instruction, the instruction address is directly written in the trace memory 306 without any compression.

In this prior art, since instruction addresses except those of branch instructions are compressed and stored in the trace memory, the capacity of the trace memory can be decreased. In addition, when compressed difference values are added starting from uncompressed data of the instruction address of a branch instruction, the instruction address can be reconstructed.

In the prior art shown in Fig. 12, however, only branch instructions are detected. If only a certain range of a program is to be repeatedly traced, as in, e.g., section trace, the original address cannot be reconstructed from compressed data. This is because the section trace start instruction address is not a branch instruction and is therefore compressed, like other instruction addresses.

Summary of the Invention

It is an object of the present invention to provide a program development support apparatus which can compress and reconstruct an instruction address even when a specific range is to be traced starting from a

specific instruction code or specific address other than a branch instruction, as in section trace, and can store an enormous amount of trace data even with a limited memory capacity.

In order to achieve the above object, according to the present invention, there is provided a program development support apparatus comprising a CPU (Central Processing Unit) for executing a target program and outputting instruction address/instruction code data, event management means for asserting and outputting a section trace start signal upon detecting that the instruction address/instruction code data from the CPU matches one of a predetermined instruction address and predetermined instruction code set as an event condition in advance, trace data generation means for, when an instruction code of the instruction address/instruction code data from the CPU is a branch instruction, or the section trace start signal from the event management means is active, outputting an uncompressed instruction address as trace data, and when the instruction address of the instruction address/instruction code data is not the branch instruction, and the section trace start signal is not active, generating a plurality of compressed instruction addresses by compressing the instruction address of the instruction address/instruction code data, and then combining the compressed instruction addresses and outputting the

compressed instruction addresses as the trace data, and a trace memory for storing the trace data from the trace data generation means.

Brief Description of the Drawings

Fig. 1 is a block diagram of a program development support apparatus according to the first embodiment of the present invention;

Fig. 2 is a circuit diagram of an instruction address/instruction code latch circuit shown in Fig. 1;

Fig. 3 is a circuit diagram of an instruction address data compression circuit shown in Fig. 1;

Fig. 4 is a circuit diagram of a trace control circuit shown in Fig. 1;

Fig. 5 is a view showing a sample program list;

Figs. 6A to 6I are timing charts showing branch instruction determination operation by the program development support apparatus shown in Fig. 1;

Fig. 7 is a view showing data in a trace memory after branch instruction determination;

Fig. 8 is a flow chart showing a trace data read procedure;

Figs. 9A to 9I are timing charts showing section trace operation by the program development support apparatus shown in Fig. 1;

Fig. 10 is a view showing data in the trace memory after section trace;

Fig. 11 is a block diagram of a program development support apparatus according to the second embodiment of the present invention; and

Fig. 12 is a block diagram of a conventional program development support apparatus.

Description of the Preferred Embodiments

The present invention will be described below in detail with reference to the accompanying drawings.

A program development support apparatus of the present invention prepares an event detection section for detecting a predetermined instruction address or predetermined instruction code externally set in advance and has a mechanism for, when an event is detected, storing in a trace memory an instruction address for storing the detected instruction address or detected instruction code without any compression.

Fig. 1 shows a program development support apparatus according to an embodiment of the present invention. Referring to Fig. 1, the program development support apparatus has an evaluation chip 1 for executing a target program and a tracer 2 for storing an execution result.

The evaluation chip 1 has a CPU 3 for actually executing the target program, and an event detection section 4 serving as an event management means. The event detection section 4 receives instruction address/instruction code data 14 from the CPU 3 and

determines whether the instruction address or instruction code matches a preset event condition. If they match, the event detection section 4 activates a section trace start signal 16 and keeps a section trace period data latch signal 15 active.

The tracer 2 has a trace data generation section 5 and a trace memory 6 for storing trace data. The trace data generation section 5 receives a clock signal 13, instruction address/instruction code data 14, section trace period data latch signal 15, and section trace start signal 16 from the evaluation chip 1. When the instruction address of the instruction address/instruction code data 14 is a branch instruction, or the section trace start signal 16 is active, the trace data generation section 5 writes an instruction address 27 in the trace memory 6 without any compression.

On the other hand, when an instruction code 28 of the instruction address/instruction code data 14 is not a branch instruction, and the section trace start signal 16 is not active, the trace data generation section 5 generates a compressed instruction address 31 corresponding to a difference value obtained by subtracting the instruction address of the preceding instruction address/instruction code data from the instruction address 27 of the current instruction address/instruction code data 14. The trace data generation section 5 also combines a plurality of

compressed instruction addresses 31 corresponding to a plurality of consecutive instruction address/instruction code data 14 in accordance with the bit width of the trace memory 6 and writes the combined data in the trace memory 6 as trace data 32.

The event detection section 4 has an event setting circuit 11 and event detection circuit 12. When an instruction address or instruction code is externally set in advance as an event condition, the event setting circuit 11 holds the active period of the data latch signal 15. Upon detecting that the set value of an instruction address or instruction code transferred from the event setting circuit 11 by an event setting data signal 17 matches the instruction address or instruction code output from the CPU 3, the event detection circuit 12 activates the section trace start signal 16 and also activates the data latch signal 15 during the period set in the event setting circuit 11. When no event condition is set, the data latch signal 15 continuously outputs the active level.

The trace data generation section 5 comprises an instruction address/instruction code latch circuit 21, instruction address data compression circuit 22, branch instruction determination circuit 23, trace control circuit 24, and 2-input OR circuit 25.

The instruction address/instruction code latch circuit 21 latches the instruction address/instruction

code data 14 output from the CPU 3 in the evaluation chip 1 on the basis of the data latch signal 15 in synchronism with the clock signal 13 and outputs the instruction address 27 and instruction code 28.

The instruction address data compression circuit 22 receives the instruction address 27 from the instruction address/instruction code latch circuit 21 and an uncompressed data selection signal 30 from the 2-input OR circuit 25. When the uncompressed data selection signal 30 is active, the instruction address data compression circuit 22 outputs the instruction address 27 as the compressed instruction address 31 without any compression. On the other hand, when the uncompressed data selection signal 30 is not active, the instruction address data compression circuit 22 outputs difference data obtained by subtracting the immediately preceding instruction address from the current instruction address as the compressed instruction address 31.

The branch instruction determination circuit 23 receives the instruction code 28 from the instruction address/instruction code latch circuit 21 and determines whether the instruction code 28 is a branch instruction. If it is determined that the instruction code is a branch instruction, the branch instruction determination circuit 23 asserts a branch instruction detection signal 29 (high level).

The trace control circuit 24 receive the compressed instruction address 31 from the instruction address data compression circuit 22 and the uncompressed data selection signal 30 from the 2-input OR circuit 25. When the uncompressed data selection signal 30 is active, the trace control circuit 24 outputs the compressed instruction address 31 as the trace data 32 without any processing. On the other hand, when the uncompressed data selection signal 30 is not active, the trace control circuit 24 combines a plurality of continuously received compressed instruction addresses 31 in accordance with the bit width of the trace memory 6 and outputs the combined data as the trace data 32, and additionally, outputs to the trace memory 6 a trace data write signal 34 for designating a write of the trace data and a trace memory address 33 for designating a storage address.

The 2-input OR circuit 25 asserts and outputs the uncompressed data selection signal 30 when at least one of the branch instruction detection signal 29 from the branch instruction determination circuit 23 and the section trace start signal 16 from the event detection section 4 is active (high level).

Details of the instruction address/instruction code latch circuit 21 will be described next with reference to Fig. 2. Referring to Fig. 2, the instruction address/instruction code latch circuit 21

holds the instruction address/instruction code data 14 by a data latch 211 and extracts the instruction address 27 and instruction code 28.

The instruction address/instruction code data 14 is basically latched by the data latch 211 in synchronism with the instruction address/instruction code data 14 in accordance with the clock signal 13 output from the CPU 3. In this case, the latch operation is ON/OFF-controlled by ON/OFF-controlling the gate of an AND circuit 212 by the data latch signal 15 from the event detection circuit 12. In the section trace, the data latch signal 15 is active only during the period from the start to the end of section trace. In the normal trace state, the data latch signal 15 always maintains the active level.

Details of the instruction address data compression circuit 22 will be described next with reference to Fig. 3. As an instruction address compression method, in the non-compression mode, the instruction address value (e.g., 32-bit length) is recorded as a base address, and in the compression mode, the difference value (e.g., 8-bit length) between the immediately preceding instruction address value and the current instruction address value is recorded as compressed instruction address data. The instruction address data compression circuit 22 comprises a preceding instruction address latch 221, subtractor 222,

and compression/non-compression switching circuit 223.

The preceding instruction address latch 221 generates difference data between the current instruction address value and the preceding instruction address value as compressed instruction address data. The subtractor 222 calculates the difference between the current instruction address and the output from the preceding instruction address latch 221 to generate compressed data. The compression/non-compression switching circuit 223 switches between the compressed data from the subtractor 222 and the instruction address data 27 as uncompressed data. When the uncompressed data selection signal 30 is asserted, the uncompressed instruction address 27 is output from the compression/non-compression switching circuit 223 as the compressed instruction address 31.

Details of the trace control circuit 24 will be described next with reference to Fig. 4. The trace control circuit 24 comprises a compressed data shift register 241, uncompressed data latch 242, trace data switching circuit 243, trace data counter 244, and 2-input OR circuit 245. The trace control circuit 24 having these components generates the trace memory address 33 to be written in the trace memory 6, the trace write signal 34 for enabling write operation, and a timing signal for finally writing the data in the trace memory 6. The trace control circuit 24 also

aligns the compressed instruction address signal 31 generated by the instruction address data compression circuit 22 to the bit width of the trace memory 6.

When the trace write signal 34 is active, the trace data counter 244 increments by one in synchronism with the leading edge of the clock signal 13. The trace memory address 33 output from the trace data counter 244 represents an address at which the trace data 32 is to be sequentially recorded in a frame of the trace memory 6. The trace memory address 33 changes in synchronism with the write timing to the trace memory 6 and is normally incremented immediately after the write in the trace memory 6 is ended. The write timing will be described later.

The operation of aligning the compressed data to the bit width of the trace memory 6 will be described next. To align compressed data having a smaller bit width than a base address to the bit width of the trace memory 6, the compressed data shift register 241 is used. When the data of the compressed instruction address 31 is compressed, the compressed data shift register 241 stores the data while sequentially shifting it until data that fill the bit width of the trace memory are stored. Referring to Fig. 4, since the data width ratio of "uncompressed data" to "compressed data" is 4 : 1, the compressed data shift register 241 sequentially stores compressed data of four instruction addresses at

maximum.

When the fourth compressed data is written in the compressed data shift register 241, an alignment completion signal 246 is asserted and recorded in the trace memory 6 through the 2-input OR circuit 245 as data of one frame. Simultaneously, the contents in the compressed data shift register 241 are cleared to prepare for storage of the next compressed data.

The instruction address 27 as uncompressed data or aligned compressed data 247 that has aligned to the bit width by the compressed data shift register 241 is selected, as the trace data to be output 32, by the trace data switching circuit 243 in accordance with a signal output from the uncompressed data latch 242. The uncompressed data latch 242 outputs a signal obtained by temporarily latching the uncompressed data selection signal 30 and adjusting its timing to the trace data switching circuit 243.

Data compression/non-compression operation for a branch instruction and normal instruction in the program development support apparatus having the above arrangement will be described next.

Fig. 5 shows a sample program list for explaining trace operation. This sample program is a partial extraction from a large-scale program. Of instructions 1 to 27, instructions 3 to 26 form a single loop so that the program repeatedly branches to

instruction 3 an arbitrary number of times in accordance with a condition branch instruction of instruction 26. The instruction address space has 32 bits. The instruction code length is 4 or 2 bytes. The trace memory uses a ring buffer having addresses (00000) to (000FF).

Figs. 6A to 6I show operation timings in determining a branch instruction. The operation shown in Figs. 6A to 6I does not use section trace and is basically the same as that of the prior art shown in Fig. 12.

When no event condition is set in the event setting circuit 11, and section trace is not used, the data latch signal 15 is always active. The instruction address/instruction code latch circuit 21 latches the instruction address/instruction code data 14 sequentially from instruction 1 in response to each clock signal 13 and outputs the instruction address 27 (Figs. 6A to 6C).

When the program progresses to instruction 26 (Fig. 5) as a branch instruction, the instruction address/instruction code data 14 output from the CPU 3 changes to instruction 3. At this time, the branch instruction determination circuit 23 determines that the branch instruction has been executed immediately before and asserts the branch instruction detection signal 29 (Fig. 6D).

The instruction address data compression circuit 22 outputs, as the compressed instruction address 31, uncompressed data of the instruction address of instruction 3 in place of the difference between immediately preceding address values, i.e., compressed data that has been output (Fig. 6E).

When the branch instruction detection signal 29 is activated, the trace control circuit 24 asserts the trace data write signal 34 and outputs it to the trace memory 6 during a 2-clock period (Fig. 6H). The trace memory address 33 is incremented to (00000) in synchronism with the first leading edge of the clock signal 13 (Fig. 6I). The instruction address of instruction 3 is written as uncompressed data at a corresponding frame address of the trace memory 6. When the write to the frame is ended, for the write to the next frame, the trace memory address 33 is incremented by one to (00001) in synchronism with the second leading edge of the clock signal 13 during the active period of the trace data write signal 34 (Fig. 6I).

After that, in accordance with instruction 4, the instruction address data compression circuit 22 calculates "instruction address of instruction 4" - "instruction address of instruction 3" and compresses the result to lower 1-byte data. In a similar manner, compressed data are generated in accordance with instructions 5, 6, and 7 and sequentially sent to the

compressed data shift register 241 of the trace control circuit 24. When the compressed data of instruction 7 is input, the compressed data shift register 241 is filled and outputs the alignment completion signal 246 (Fig. 6G). The trace control circuit 24 generates the trace data write signal 34 in accordance with the alignment completion signal 246 and writes in the trace memory 6 the trace data 32 obtained by combining the compressed data of four instructions 4 to 7 (Figs. 6G and 6H). After that, the trace memory address 33 is incremented by one to (00002) in synchronism with the leading edge of the clock signal 13. This operation is continued until the next branch instruction is detected.

Fig. 7 shows the data state in the trace memory 6 after branch instruction determination. This data state is the execution result of the sample program recorded in the trace memory 6 by the above operation.

In the trace data stored in the trace memory 6, data corresponding to instruction 3 has a compression flag "0" which represents that the instruction address can be read out as uncompressed data. For each of the remaining instructions, since an increment (matching the number of bytes of instruction code) from the immediately preceding instruction address is stored, a reconstructed address can be obtained by reading out the instruction address and adding it to the preceding instruction address.

Trace data read operation will be described next in detail with reference to Fig. 8.

Steps S1 to S3 correspond to initialization of the read operation. In step S1, the read pointer is set to the start frame of the trace memory 6. In step S2, the instruction address at the start frame is set as a base address. It is determined in step S3 whether the compression flag is "1". If NO in step S3, the flow advances to step S4. If YES in step S3, the flow advances to step S5.

In step S4, the trace data is latched from the start frame of the trace memory 6, and the instruction address at the uncompressed frame is set as a base address. In step S6, the instruction code of the address is read from the base address, and the instruction is displayed. In step S7, the read pointer is incremented, and the next trace frame is read out.

If YES in step S3, the frame data is compressed data. In step S5, an offset is added to the base address to set a new base address. The flow advances to step S6 to perform the same operation as described above. The instruction addresses are reconstructed according to the above procedure to obtain the reconstructed addresses shown in Fig. 7.

Operation using section trace will be described next with reference to Figs. 9A to 9I.

In section trace, trace is performed only for

a specific range of a program. As a detailed example, operation of performing trace only for the range of four steps from instruction 4 to 7 in the sample program list shown in Fig. 5, where the data is written, will be described.

The instruction address/instruction code data 14 output from the CPU 3 is sampled by the data latch 211 of the trace data latch circuit 21 only during the period when instructions 4 to 7 are being executed on the basis of the active period of the data latch signal 15, that is set in the event setting circuit 11. For this reason, as the instruction address 27, only the instruction addresses of instructions 4 to 7 are output from the instruction address/instruction code latch circuit 21 to the instruction address data compression circuit 22 (Fig. 9C).

At this time, the section trace start signal 16 representing the section trace start position is asserted during the 2-clock period from the start of execution of instruction 4 (Fig. 9D). The section trace start signal 16 is input to the 2-input OR circuit 25 together with the branch instruction detection signal 29. When the section trace start signal 16 is asserted (high level), the uncompressed data selection signal 30 output from the 2-input OR circuit 25 is asserted independently of whether the instruction is a branch instruction. As a result, the instruction address of instruction 4 is

output during the low level period of the clock signal 13 as uncompressed data.

Generation of compressed data of instructions 5 to 7 and combination to trace data are the same as in Figs. 6A to 6I, and a description thereof will be omitted.

Fig. 10 shows the data state in the trace memory after section trace. This data state is the execution result of section trace of the sample program recorded in the trace memory 6 by the above operation. Data corresponding to instruction 4 as trace data has a compression flag "0" which represents that the instruction address is uncompressed. The trace data read in section trace is also the same as in Fig. 8.

Fig. 11 shows a program development support apparatus according to the second embodiment of the present invention. The basic arrangement of the program development support apparatus shown in Fig. 11 is the same as in the first embodiment shown in Fig. 1 except that a frame address comparison circuit 26 is added, and a 3-input OR circuit 25a is used in place of the 2-input OR circuit 25.

The frame address comparison circuit 26 receives an event setting data signal 17 and trace memory address 33, and when predetermined portions of the two data match, asserts a frame match signal 35. The 3-input OR circuit 25a receives the frame match

signal 35 in addition to a branch instruction detection signal 29 and section trace start signal 16 and, when at least one of these signals is active, asserts and outputs an uncompressed data selection signal 30.

When, e.g., a value that matches the trace memory address 33 at a predetermined frame interval is set in an event setting circuit 11 as an event condition, uncompressed trace data can be embedded at an arbitrary frame. In this embodiment, even when the section trace range exceeds the maximum number of frames of the trace memory (in the example used for explanation, the frames of the trace memory are 00000 to 000FF), uncompressed data can be embedded into trace data. Hence, even in this case, the original instruction address can be reconstructed from compressed trace data using the uncompressed data as a base instruction address.

As has been described above, according to the present invention, an instruction address, which is not to be compressed and is to be used as a base address in reconstructing compressed trace data, can be externally set in the event detection section. As a consequence, even when the trace data contains no branch instruction, as in section trace, the base address as the base of reconstruction can be left in the trace data, and the instruction address can be correctly reconstructed.

Not only in section trace but also in trace of DMA (Direct Memory Access) or interrupt of the CPU,

address information of an instruction for changing the program execution order, although it contains no branch instruction, or an instruction for performing a bus access different from the CPU operation is set. In this case as well, since an instruction address can be stored in the trace memory as uncompressed data and used as a base address in reconstructing the instruction address, the instruction address can be correctly reconstructed.